

Worklog Field Plugin

This page will help you to build your own Worklog Field.

Prerequisites

You will need to follow below prerequisites :

- Have some basis knowledge on how to build a JIRA Plugin. Take time to read [Building JIRA add-ons](#).
- Understand that :
 - Add-Ons extend JIRA by using some of provided [Plugin Module Type](#),
 - An AddOn can also provide new Module Type by using the [Module Type Plugin Module](#),

Module Types provided Worklog Fields

To allow to develop your own **Worklog Fields**, Minyaa Time provides 2 disctints Module Types :

- **<plugin-model />** : It allows a plugin to define additional OFBiz Entity Configurations.
- **<worklog-field />** : It allows to define a new Worklog Field

Your Worklog Field Plugin step by step ..

Step 1. Create the plugin skeleton

Here, you will create your plugin skeleton with your preferred solution :

- by using the Atlassian SDK : See the [Atlassian's documentation](#),
- or manually, if you are aware of its content

Step 2. Update your Maven POM ...

... related the dependencies

In terms of dependencies, you will need to append Minyaa Time (here for JIRA 7.3.x using Minyaa Time 1.16)

```
<project ...>
  <dependencies>
    <dependency>
      <groupId>com.atlassian.jira</groupId>
      <artifactId>jira-api</artifactId>
      <version>${jira.version}</version>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>fr.alkaes.minyaa</groupId>
      <artifactId>jira-plugin-minyaa-time</artifactId>
      <version>${minyaa.time.version}</version>
      <scope>provided</scope>
    </dependency>
  </dependencies>
  ...
  <properties>
    <jira.version>7.3.0</jira.version>
    <jira.data.version>${jira.version}</jira.data.version>
    <minyaa.time.version>7300.1.16</minyaa.time.version>
  </properties>
</project>
```

... related the Maven JIRA Plugin

On this page:

- [Prerequisites](#)
- [Module Types provided Worklog Fields](#)
- [Your Worklog Field Plugin step by step ..](#)
 - [Step 1. Create the plugin skeleton](#)
 - [Step 2. Update your Maven POM](#)
 - ...
 - [... related the dependencies](#)
 - [... related the Maven JIRA Plugin](#)
 - [Step 3. Create your Worklog Fields Classes](#)
 - [Interface WorklogField](#)
 - [Sample : Chargeable](#)
 - [Validation capacities](#)
 - [Step 4. Configure your OFBiz Entity](#)
 - [Step 4. Configure your Modules in Atlassian Descriptor](#)
 - [Worklog Fields Entity Model Declaration](#)
 - [Att](#)
 - [Ele](#)
 - [Sa](#)
 - [Worklog Fields Classes Declaration](#)
 - [Att](#)
 - [Sa](#)
 - [I](#)
 - [Step 5. Build and Deploy](#)
- [Sample Plugin : jira-plugin-minyaa-worklogfields](#)

Since the components implemented in your plugin will have to be loaded by Minyaa Time, you will have to make them exportable.

Here, we assume that your Worklog Fields have **com.yourcompany.jira.worklogfields** as package.

```
<project ...>
  <build>
    <plugins>
      <plugin>
        <groupId>com.atlassian.maven.plugins</groupId>
        <artifactId>maven-jira-plugin</artifactId>
        <version>${amps.version}</version>
        <extensions>true</extensions>
        <configuration>
          <productVersion>${jira.version}</productVersion>
          <productDataVersion>${jira.data.version}<
/productDataVersion>
            <instructions>
              <Bundle-SymbolicName>${project.groupId}.${project.
artifactId}</Bundle-SymbolicName>
              <Spring-Context>*;timeout:=60</Spring-Context>
              <Import-Package>
                *;resolution:=optional
              </Import-Package>
              <Export-Package>com.yourcompany.jira.worklogfields.
*</Export-Package>
            </instructions>
          </configuration>
        </plugin>
      ...
    </plugins>
  </build>
  ...
  <properties>
    <amps.version>6.2.2</amps.version>
    <jira.version>7.3.0</jira.version>
    <jira.data.version>${jira.version}</jira.data.version>
    ...
  </properties>
</project>
```

Step 3. Create your Worklog Fields Classes

Then, you will have to develop your first **Worklog Field** class ...

Interface WorklogField

Minyaa Time provides the below WorklogField interface

```
public interface WorklogField {

  /**
   * @return the Module Descriptor for Worklog Field
   */
  WorklogFieldModuleDescriptor getDescriptor();

  /**
   * @return the Key of the field as defined in its descriptor
   */
  public String getKey();

  /**
   * @return the Name of the OFBiz Entity where the field will be stored
   */
  public String getEntityName();

  /**
   * @return the Name of the field as it will be used in the OFBiz Entity
```

```

    */
    public String getFieldName();

    /**
     * @return the Id of the field as it will be used in the HTML Form
     */
    public String getId();

    /**
     * @return a generated Complete Key used to identify the field among
all fields provided by any plugin
     */
    public String getCompleteKey();

    /**
     * @return the key of the field as defined in its descriptor
     */
    public WorklogField newWorklogField();

    /**
     * @return the display name as it will be displayed in the GUI (No I18n
supported yet)
     */
    public String getDisplayName();

    /**
     * @return the value as it is passed from HTML Form
     */
    public Object getValue();

    /**
     * Sets the value of field
     */
    public void setValue(Object value);

    /**
     * @return the default value of (user uninitialized) field
     */
    public Object getDefaultValue();

    /**
     * Performs syntax and semantic (eg. hours between 0 and 23) field
validation.
     * @param worklogFields Map of all passed Worklog Fields. It may be
required to perform validation against other passed Worklog Fields
     * @return true if field validated OK, false otherwise
     */
    public boolean validate(final Map<String,WorklogField> worklogFields);

    /**
     * NOTE YET USED !!
     * Calculates field visibility depending on its value or/and
potentially other field values
     * @param worklogFields Map of all passed Worklog Fields. It may be
required to evaluate other passed Worklog Fields
     * @return true if hidden, false otherwise
     */
    public boolean hide(final List _worklogFields);

    /**
     * @return display type is the input control type (text, select, radio,
checkbox, ...) to use in the HTML Form
     */
    public String getDisplayType();

    /**
     * @return The Native Value is the Raw value as it has to be stored by
OFBiz
     */
    public Object getNativeValue();

```

```

    public void formatForDisplay();

    public Worklog getWorklog();

    public void setWorklog(final Worklog _worklog);

    public String getValidationError();

}

```

and also an abstract implementation : ***fr.alkaes.myaatm.worklog.fields.AbstractWorklogField***.

When you will implement your own Worklog Field, you WILL HAVE TO use it.

Sample : Chargeable

Here the source of ***fr.alkaes.myaatm.worklog.fields.impl.Chargeable***, a Worklog Field provided by default in Minyaa Time (just a sample for the moment).

```

package fr.alkaes.myaatm.worklog.fields.impl;
import java.util.List;
import fr.alkaes.myaatm.worklog.fields.AbstractWorklogField;
/**
 * @FCQN fr.alkaes.myaatm.worklog.fields.impl.Chargeable
 * @author vthoule (Alkaes Consulting)
 * @since [6400/7300].1.16
 * @version [6400/7300].1.16
 * @description
 */
public class Chargeable extends AbstractWorklogField {

    private static final long serialVersionUID = 1L;

    public Chargeable() {
        // Here is defined the list of allowed values.
        allowedValues = new String[] { "0", "1", "false", "true" };
    }

    /** @return the Display Name, as it will be displayed in the HTML Form
    */
    public String getDisplayName() {
        return "Chargeable";
    }

    /** @return Default Value*/
    public Object getDefaultValue() {
        return "false";
    }

    /** Return if the field has be shown or not (NOT Yet used)
    */
    public boolean hide(List iWorklogFields) {
        return false;
    }

    /** @return HTML type of field */
    public String getDisplayType() {
        return "checkbox";
    }
}

```

Validation capacities

If needed, you will be able to validate your field against :

- Worklog : The current Worklog is referenced in the Field
- and other Worklog Fields : Map of other Worklog Field is passed to the validate() method with a key based on "***entityName:fieldName***"

```

public boolean validate(Map<String, WorklogField> iWorklogFields) {
    // validate
    boolean valid = true;

    // Retrieve Timespent of the Worklog
    Long timeSpent = worklog != null ? worklog.getTimeSpent() : null;
    // If timespent is specified
    if (timeSpent != null && timeSpent.longValue() > 0) {
        WorklogField otherWorklogField = iWorklogFields.get
("MyWorklogDetails:otherWorklogField");

        if (((Number)getNativeValue()).doubleValue()*3600 == timeSpent.
doubleValue()) {
            valid = valid && ((String)otherWorklogField.getValue()).
equalsIgnoreCase((String)otherWorklogField.getDefaultValue());
        }
        if (!valid) {
            validationError = "otherWorklogField cannot be selected for
this value.";
            return valid;
        }
    } else {
        valid = super.validate(iWorklogFields);
    }

    return valid;
}

```

Create also all needed Worklog Fields.

Step 4. Configure your OFBiz Entity

When you have defined all your Worklog Fields, you will have to prepare the OFBiz configuration and also be able to store their values.

It will require 2 distinct XML files :

- The entity group file where your additional entities will be listed.
- The entity model file where each of this entity will be configure for each fields

NB: *You are able to store your Worklog Field in one or many entities as you want,*

Here is the sample configuration for 2 Worklog Fields stored in 1 Entity

```

• <?xml version="1.0" encoding="UTF-8"?>
  <!DOCTYPE entitygroup PUBLIC "-//OFBiz//DTD Entity Group//EN"
    "http://oss.org.cn/ossdocs/applications/ofbiz/ofbiz-2.1.1-docs
    /website/dtds/entitygroup.dtd">
  <entitygroup>

    <!-- WorklogType data -->
    <entity-group group="default" entity="SampleWorklogField"/>

  </entitygroup>

```

- ```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE entitymodel PUBLIC "-//OFBiz//DTD Entity Model//EN"
"http://oss.org.cn/ossdocs/applications/ofbiz/ofbiz-2.1.1-docs
/website/dtds/entitymodel.dtd">

<entitymodel>
 <title>Entity Model for Sample Worklog Fields</title>
 <description>Entities added to extend Worklog Features<
/description>
 <copyright>Copyright (c) 2017-2017 Alkaes Consulting<
/copyright>
 <author>Vincent Thoule</author>
 <version>1.16</version>

 <entity entity-name="SampleWorklogField" table-name="
sampleworklogfields" package-name="">

 <!-- Mandatory Field used to identify the Worklog -->
 <field name="id" type="numeric" />

 <!-- "Chargeable" stored as small string -->
 <field name="chargeable" type="short-varchar" />

 <!-- "Chargeable Hours" stored as Double (Number) -->
 <field name="chargeablehours" type="floating-point"
/>

 <!-- Define Id as the Primary Key -->
 <prim-key field="id" />

 <!-- Define a relation 0,1 to 1 with Worklog -->
 <relation type="one" title="Is" rel-entity-name="
Worklog">
 <key-map field-name="id" rel-field-name="id"
/>
 </relation>
 </entity>
</entitymodel>

```

## Step 4. Configure your Modules in Atlassian Descriptor

When you have implemented all needed Worklog Fields and defined how they will stored in database, you will have to complete the configuration of the Atlassian Descriptor (***atlassian-plugin.xml***) of your plugin.

You will have to use the 2 module-types previously mentioned ...

### Worklog Fields Entity Model Declaration

The **<plugin-model />** module type allows you to declare the OFBiz Configuraiton files. They will be injected in OFBiz as soon as the plugin is enabled.

#### Attributes

| Name | Required                                                                            | Description                                       |
|------|-------------------------------------------------------------------------------------|---------------------------------------------------|
| key  |  | The unique identifier of the plugin module.       |
| name |                                                                                     | A simple Human readable name of the plugin module |

#### Elements

| Name | Attributes | Required | Description |
|------|------------|----------|-------------|
|------|------------|----------|-------------|

|          |          |   |                                                                                                                                                                                                        |
|----------|----------|---|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| resource | type     | ✓ | Only <b>type="ofbiz"</b> is expected                                                                                                                                                                   |
| resource | name     | ✓ | Two name are supported : <ul style="list-style-type: none"> <li>• <b>name="entity"</b> for the OFBiz entity model file</li> <li>• <b>name="entitygroup"</b> for the OFBiz entity group file</li> </ul> |
| resource | location | ✓ | The location of the OFBiz configuration file                                                                                                                                                           |

## Sample Declaration

Here is the declaration of **SampleWorklogField**.

```
<plugin-model key="SampleWorklogField" name="Sample Worklog Fields" >
 <resource type="ofbiz" name="entity" location="fr/alkaes/myaatm
/entitydefs/sample/entitymodel.worklogfields.xml"/>
 <resource type="ofbiz" name="entitygroup" location="fr/alkaes
/myaatm/entitydefs/sample/entitygroup.worklogfields.xml"/>
</plugin-model>
```

## Worklog Fields Classes Declaration

The **<worklog-field />** module type allows you to declare your Work Field classes. They will be injected in the **Log Work Page** and **Log Work Issue Panel** as soon as the plugin is enabled.

### Attributes / Elements

Name	Required	Description
key	✓	The unique identifier of the plugin module. It is used to build the Id for the HTML Form in Log Work Page
name		A simple Human readable name of the plugin module
weight		Determines the order in which Worklog Field appear. The 'lightest' weight is displayed first, the 'heaviest' weights sink to the bottom.
class	✓	The Java class which implements this Worklog Field. The class you need to provide must inherit from <b>fr.alkaes.myaatm.worklog.fields.AbstractWorklogField</b> and implement <b>fr.alkaes.myaatm.worklog.fields.WorklogField</b>.   The provided Java class can be provided :       • by your current plugin     • but also by any other plugins       It means that you can reuse a Worklog Field class provided by another plugin as soon as it is stored in its own OFBiz entity (see <b>field-name</b> and <b>entity-name</b>).
field-name	✓	The name of the field as it is defined in the OFBiz Entity Model configuration file.
entity-name	✓	The name of the OFBiz Entity as it is defined in the OFBiz Entity Model configuration file.
i18n-description-key		The localisation key for the human-readable description of the Worklog Field.

## Sample Declaration

```
<worklog-field weight="1" key="chargeable" name="Chargeable"
class="fr.alkaes.myaatm.worklog.fields.impl.Chargeable"
field-name="chargeable" entity-name="SampleWorklogField"
i18n-description-key="yourcompany.worklog.field.chargeable.
description" />
```

## I18n Resource

If you have defined a i18n-description-key into your Worklog Field declaration, you will have to provide an l18n Resource Property File declaration...

```
<resource type="i18n" name="i18n" location="com.yourcompany.customworklog.i18n.i18n" />
```

and the related l18n Resource Property File.

```

Your Worklog Fields

yourcompany.worklogfield.chargeable.description=Check it, if the related
spent time is chargeable.
```

## Step 5. Build and Deploy

As soon as the declaration of Worklog Fields and their related Storage Model are done, you have to compile the plugin.

It will be ready for deployment with Minyaa Time plugin as dependency.

## Sample Plugin : jira-plugin-minyaa-worklogfields

Do not hesitate to download the sample plugin in expected version

Version	JIRA Compatibility
<b>6400.1.0</b>	from 6.4.12 to 8.0.0

Find below the available versions :

File	Modified
File <a href="#">jira-plugin-minyaa-worklogfields-6400.1.0.obr</a>	Feb 14, 2019 by Vincent Thoulé
Java Archive <a href="#">jira-plugin-minyaa-worklogfields-6400.1.0-sources.jar</a>	Feb 14, 2019 by Vincent Thoulé
Java Archive <a href="#">jira-plugin-minyaa-worklogfields-6400.1.1-sources.jar</a>	Feb 28, 2019 by Vincent Thoulé
File <a href="#">jira-plugin-minyaa-worklogfields-6400.1.1.obr</a>	Feb 28, 2019 by Vincent Thoulé
Java Archive <a href="#">jira-plugin-minyaa-worklogfields-7000.1.1-sources.jar</a>	Feb 28, 2019 by Vincent Thoulé
File <a href="#">jira-plugin-minyaa-worklogfields-7000.1.1.obr</a>	Feb 28, 2019 by Vincent Thoulé
Java Archive <a href="#">jira-plugin-minyaa-worklogfields-7200.1.1-sources.jar</a>	Feb 28, 2019 by Vincent Thoulé
File <a href="#">jira-plugin-minyaa-worklogfields-7200.1.1.obr</a>	Feb 28, 2019 by Vincent Thoulé
Java Archive <a href="#">jira-plugin-minyaa-worklogfields-7300.1.1-sources.jar</a>	Feb 28, 2019 by Vincent Thoulé

File jira-plugin-minyaa-worklogfields-7300.1.1.obr	Feb 28, 2019 by Vincent Thoulé
Java Archive jira-plugin-minyaa-worklogfields-80000.1.1-sources.jar	Feb 28, 2019 by Vincent Thoulé
File jira-plugin-minyaa-worklogfields-80000.1.1.obr	Feb 28, 2019 by Vincent Thoulé

[Download All](#)